Building a robot powered with RIOT OS

Gilles DOFFE - 09/13/2018

Savoir-faire Linux France
14 rue Dupont des Loges 35000 RENNES
contact@savoirfairelinux.com

Cortex

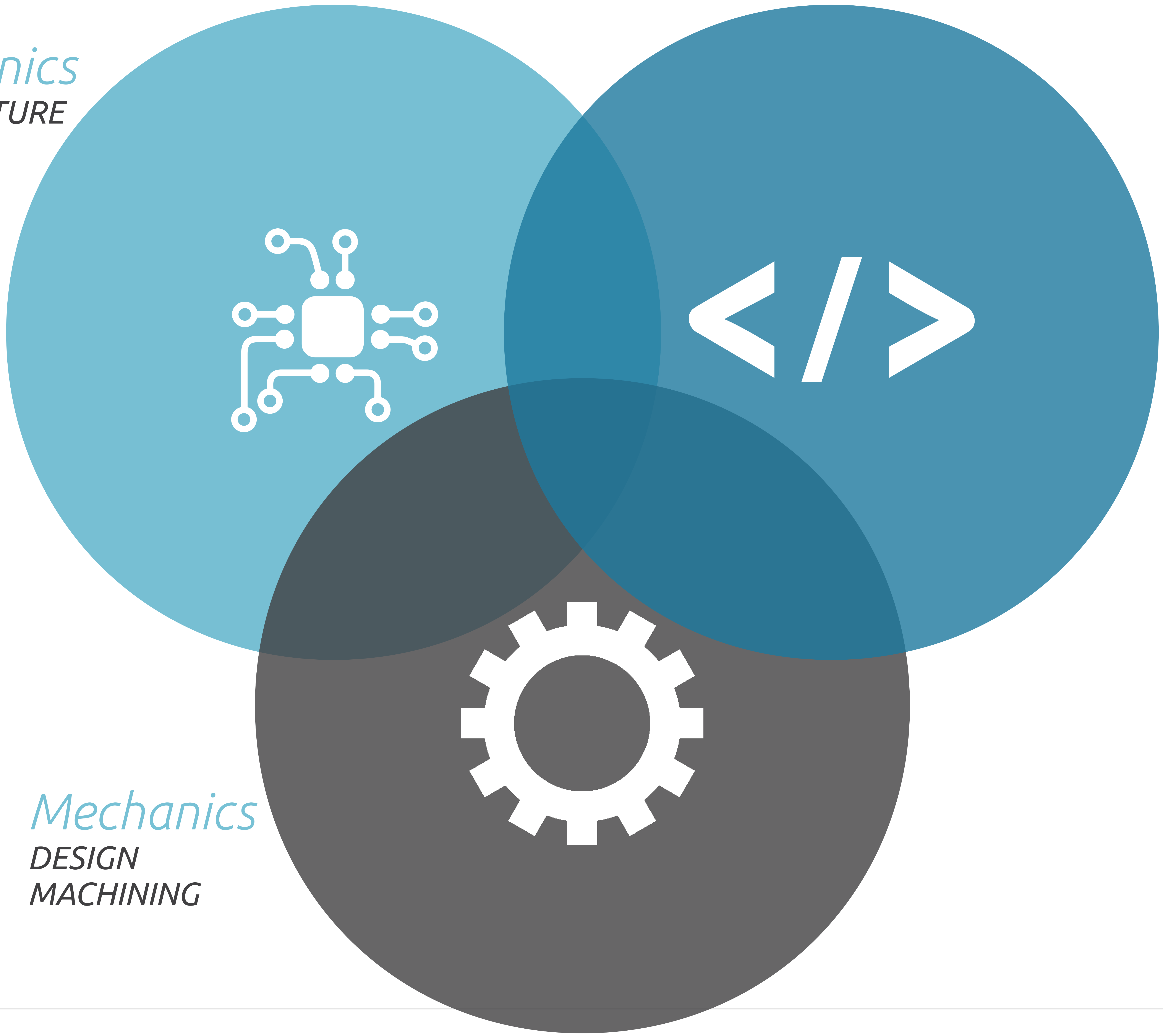
Cortex is a robot built for the French Robotic Cup 2018, qualificative phase of the Eurobot contest.

This event occurs each year in May in La-Roche-Sur-Yon, in west of France.

THANKS

- Savoir-faire Linux
- COGIP TEAM
  - Yannick Gicquel : electronics & software
  - Stephen Clymans : software
  - Cédric Wolff : mechanics
  - Gilles Doffe : machining & software
  - Estelle Taupin : logistics
  - Pierre Delignieres, Axel & Robin Doffe : secondary Bee robot using Lego
- Partners:
  - LABO Cesson
  - CEMA Technologie

Robotics : Multi area of science

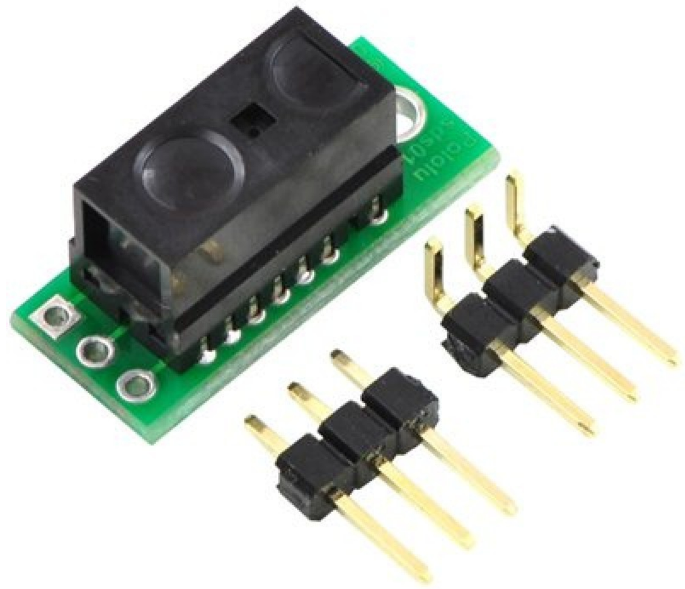


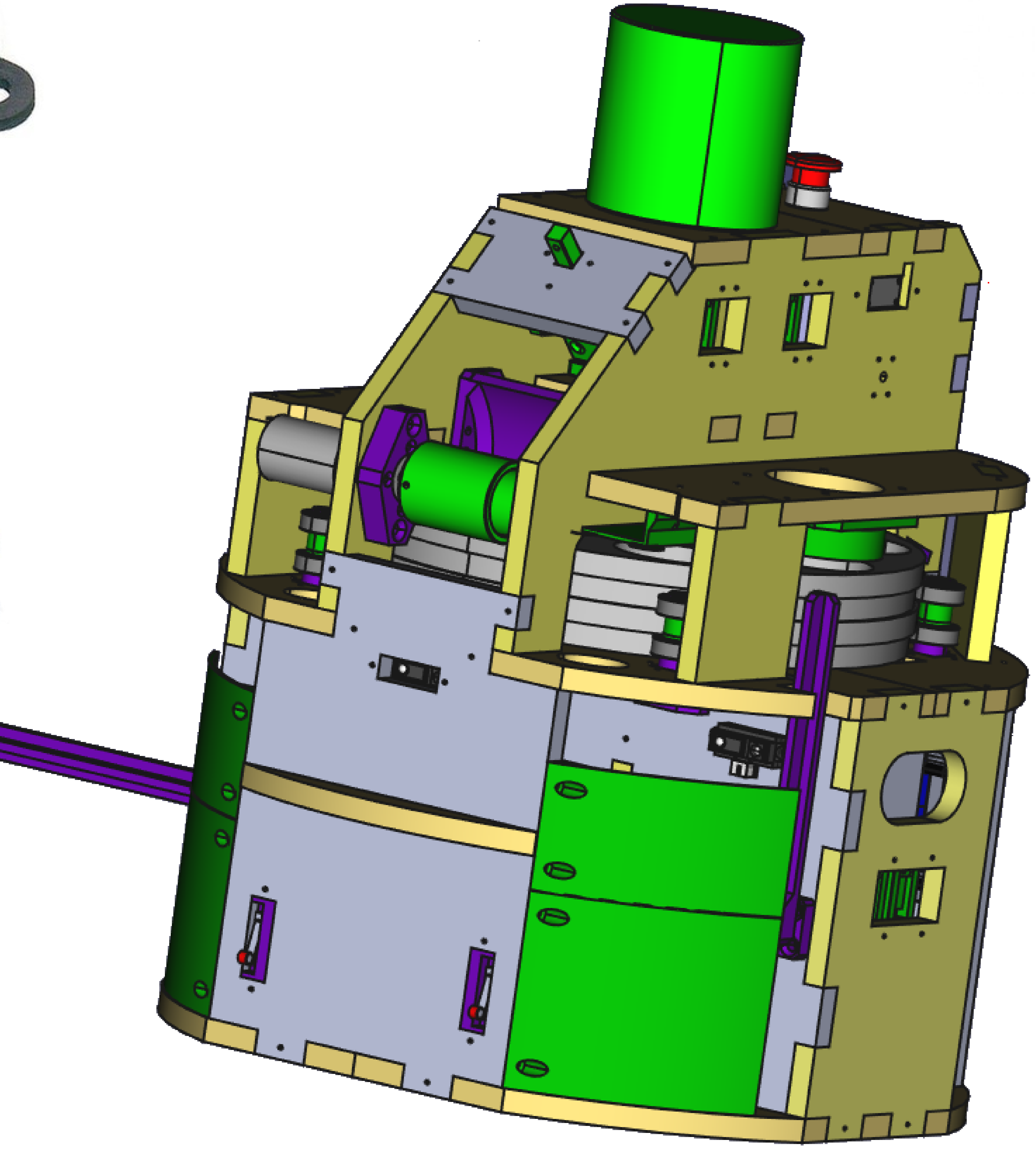
Electronics
ARCHITECTURE
DESIGN

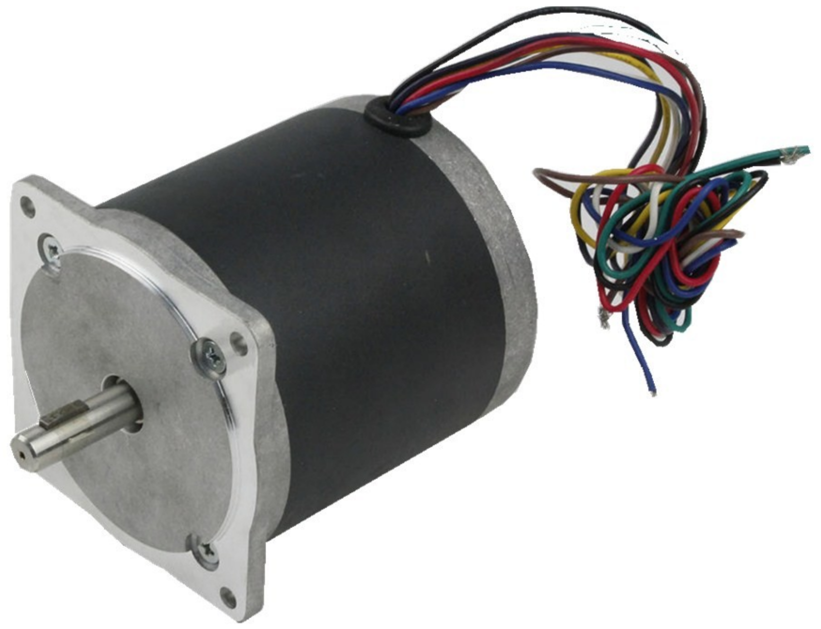
Software
ARCHITECTURE
DEVELOPMENT
INTEGRATION
VALIDATION

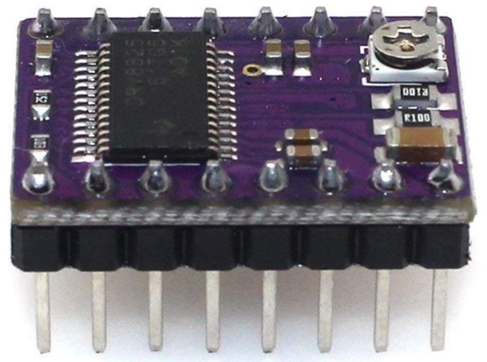
Mechanics
DESIGN
MACHINING

CORTEX



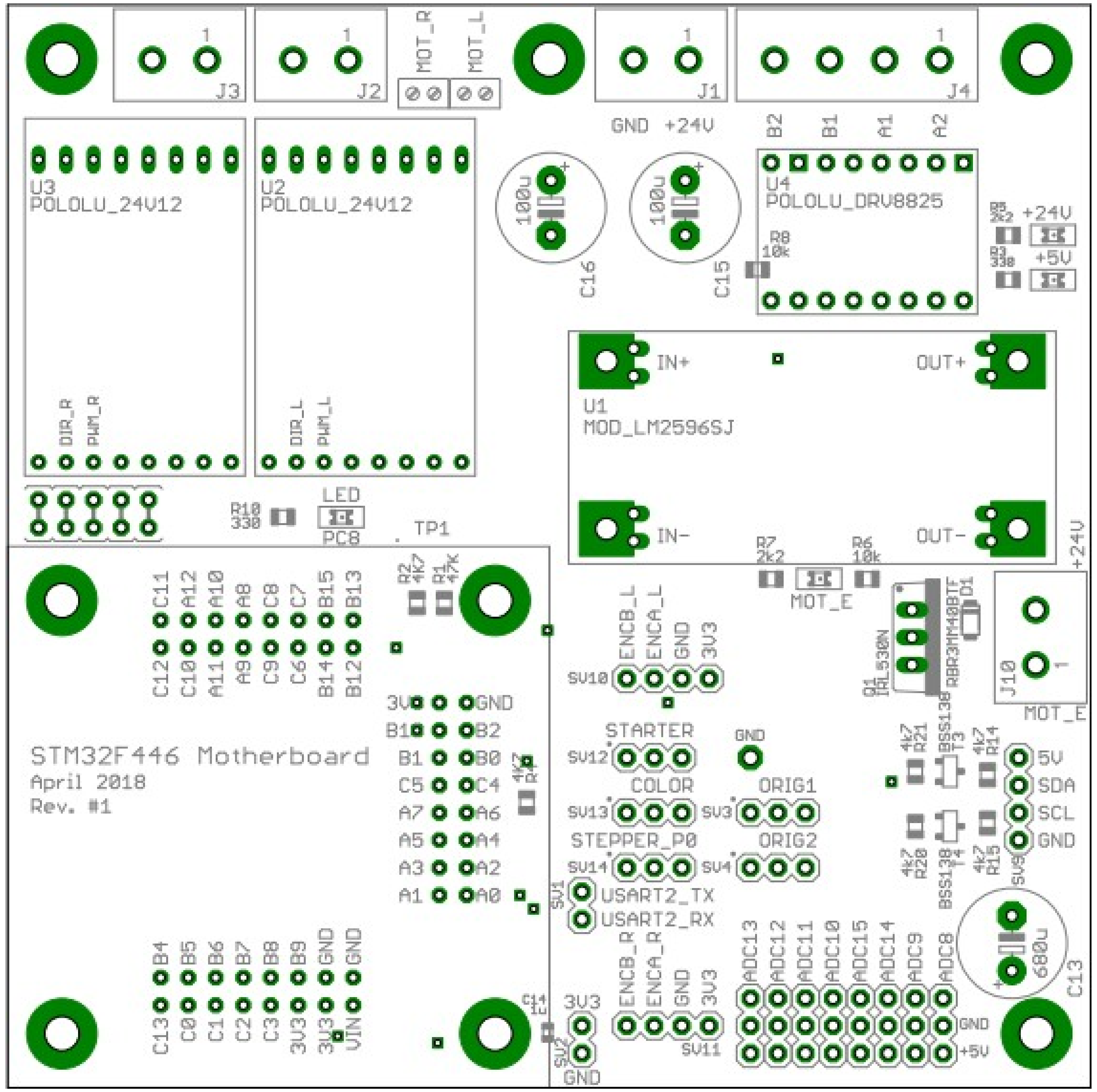


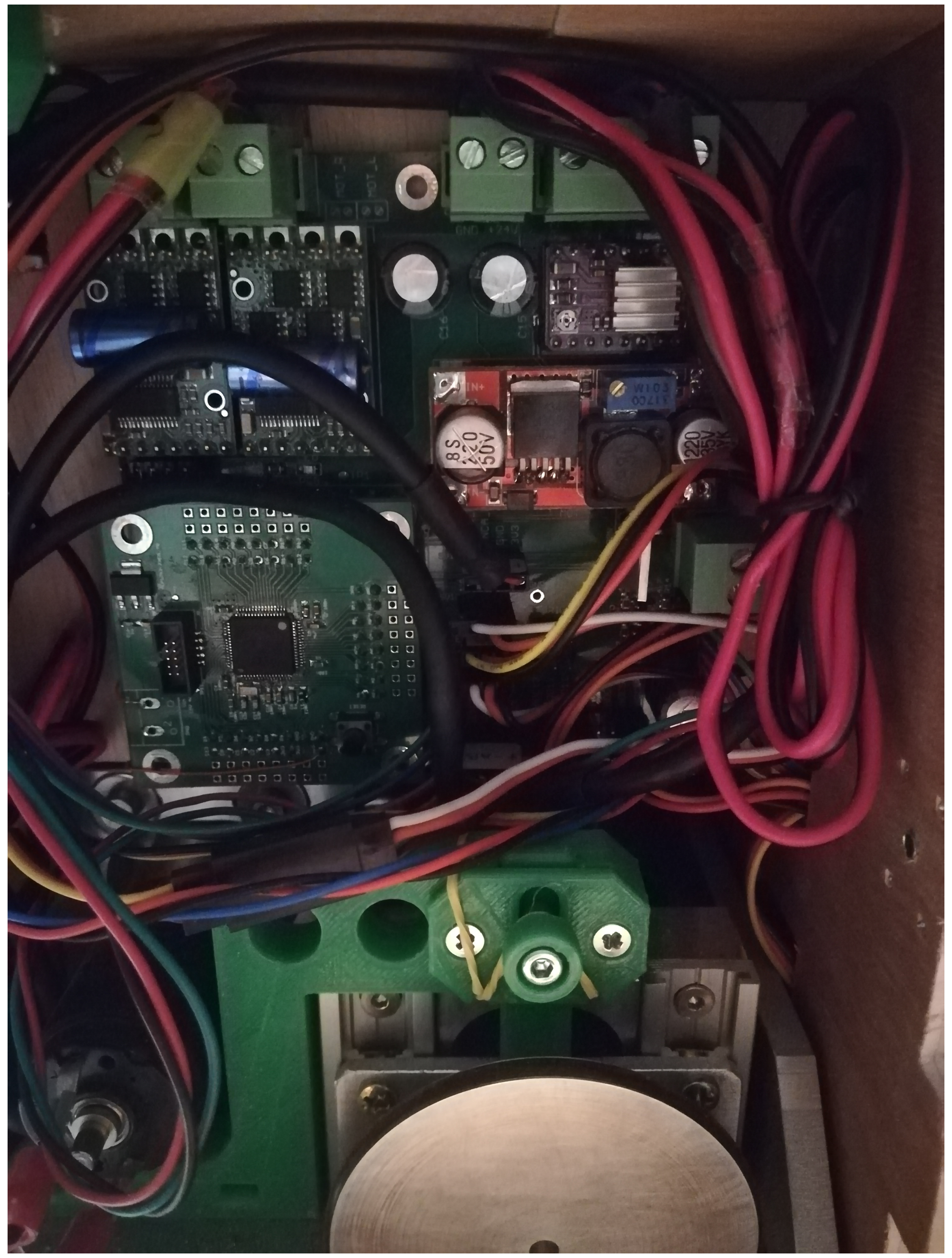




MCU

- STM32F446
  - ARM Cortex M4
  - Frequency : 180MHz
- Peripheral used :
  - 3 PWM
  - 2 QDEC
  - 2 UART
  - I2C
  - 8 ADC
  - GPIOs



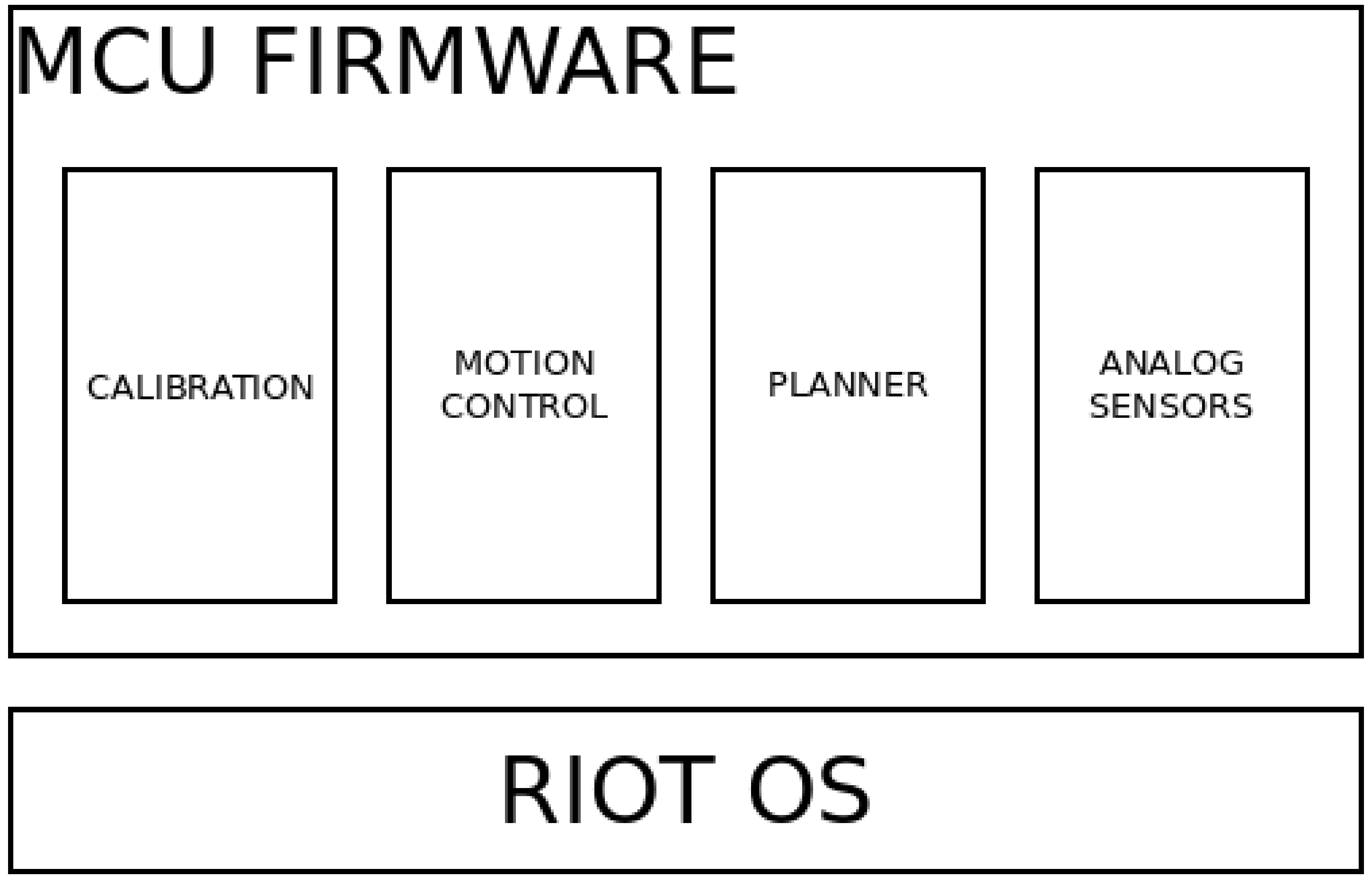


Architecture and scheduling

Cortex runs 4 threads with cooperative default scheduling.

Architecture

- Three threads sorted by decreasing priorities:
  - Motion control
  - Planner
  - Analog sensors
- Optional calibration thread
  - Calibrate servomotors and sensors
  - Step by step debugging
  - Tune PID parameters
  - It uses getchar(), which introduce blocking calls



Scheduling

- Cooperative : no systick
- All threads are fired in a sequential way in priority order
- Each thread allows the next one to be run once it finish
- Everything is done in one period of 20ms, hoping it works…

- We need to turn RIOT into a preemptive real-time OS
  - STM32 already has a hardware systick timer :
    - PR #9332 shows an example
    - Rework this PR to turn it into a generic API
  - Rework our robot source code accordingly (mutex, priority inversion, …)

Motion control

Cortex is propelled using 2 differential wheels.

The motion control algorithm makes sure the robot rolls straight using a quad PID corrector.

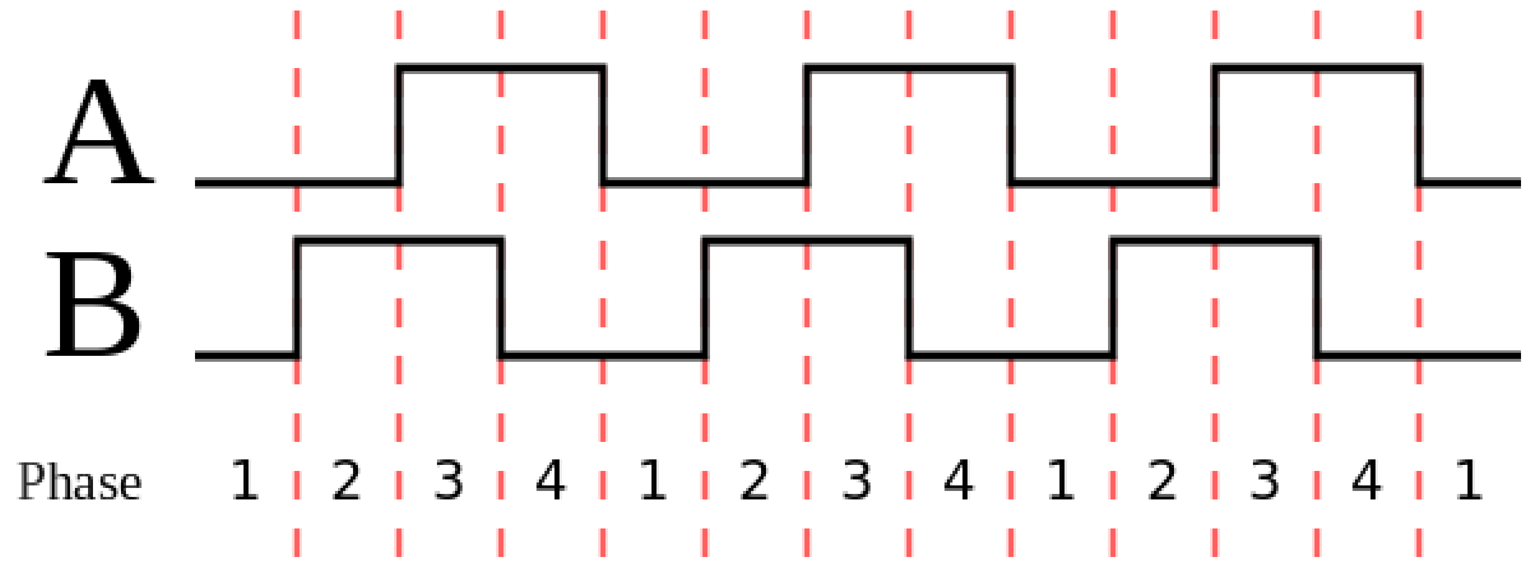
Motion control needs

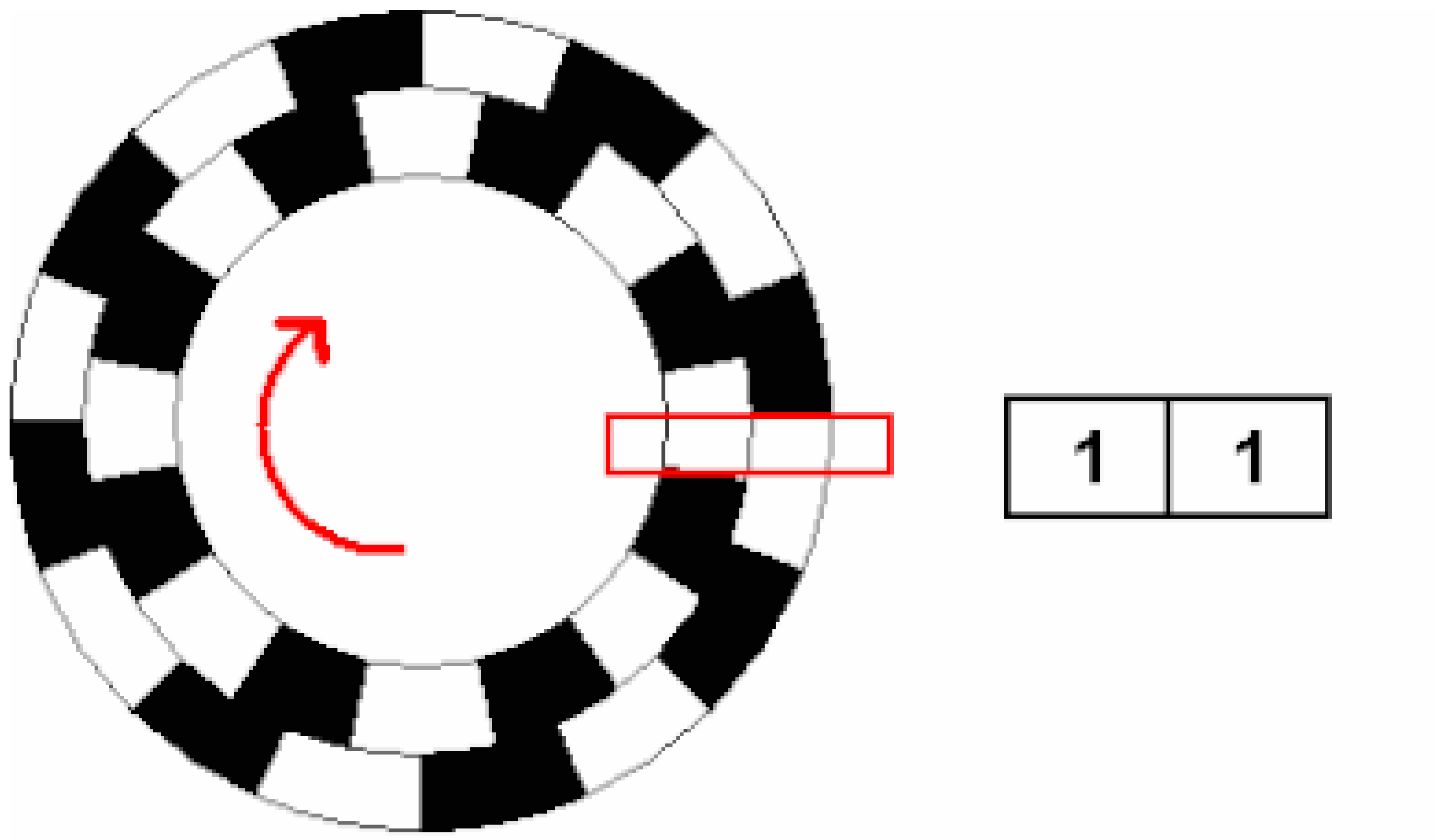
- Driving 2 DC gearhead motors
  - DC motors driver has been developed for RIOT OS (incoming PR).
  - Can drive several types of H-bridge drivers
  - To be tested : brushless motor and stepper motor in continuous mode
- Measuring distance from incremental encoders (phase quadrature)
  - QDEC peripheral driver (PR #8482 merged).
- Motion simulation
  - Problem : One robot for several developers
  - Solution : Implement PWM and QDEC for native architecture

Motion mechanic base

QDEC driver API

- Count clockwise or counter-clockwise
- Manage 3 modes :
  - QDEC_X1
  - QDEC_X2
  - QDEC_X4
- Supported architectures :
  - STM32 (hardware timer feature)
  - Native (mainly for simulation purpose)
- Other candidate architecture :
  - Atmel AVR atxmega





Source: Wikipedia

Motor driver API

- Features :
  - Support most of H-bridge hardware drivers
  - Support several motors by hardware drivers
  - Direction (CW and CCW)
  - Brake if available
  - Speed control (using PWM)
- Multi-arch driver
  - MCU requirements :
    - PWM driver
    - GPIO support
- Incoming PR :)

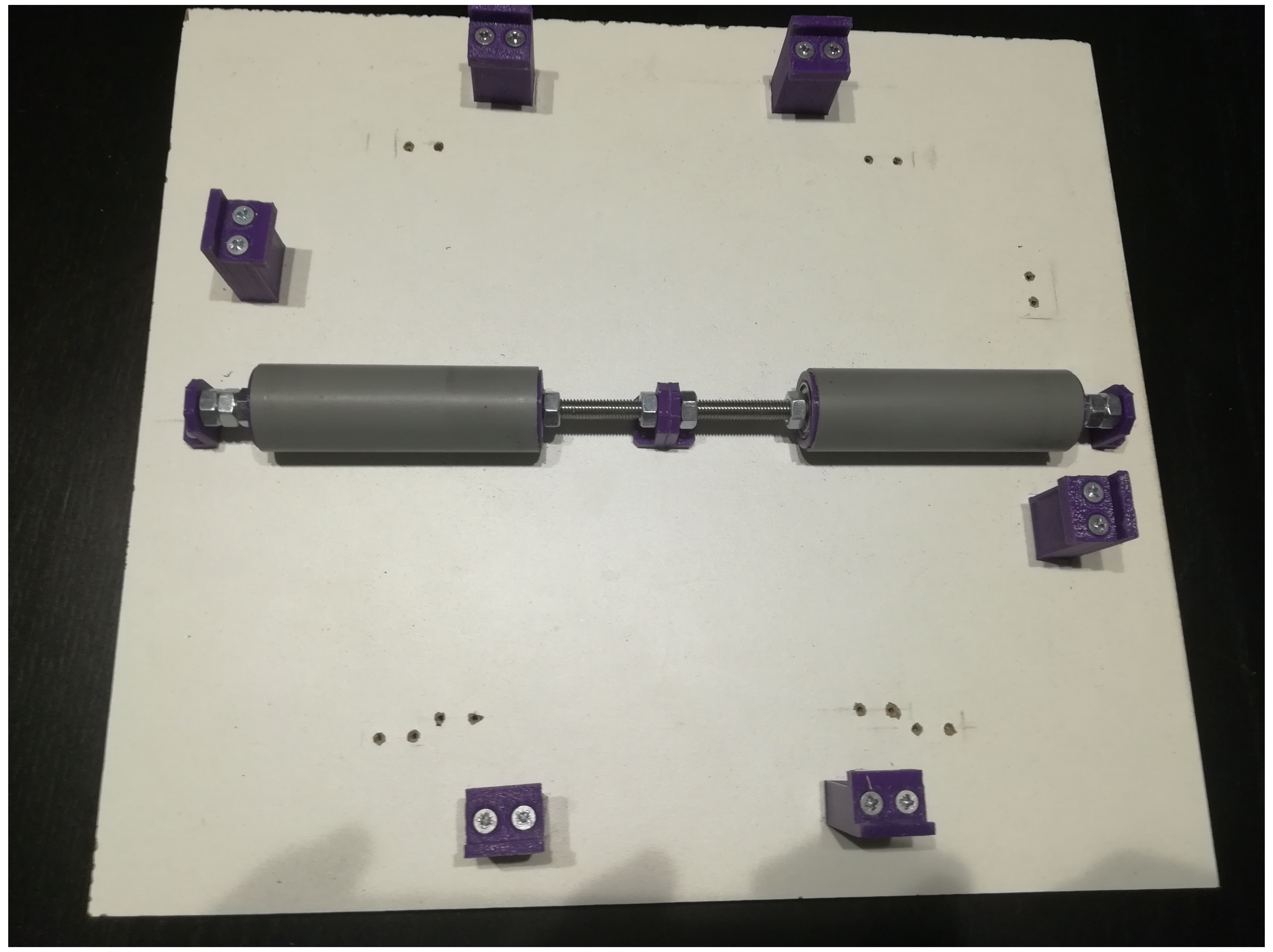
Motion control simulation

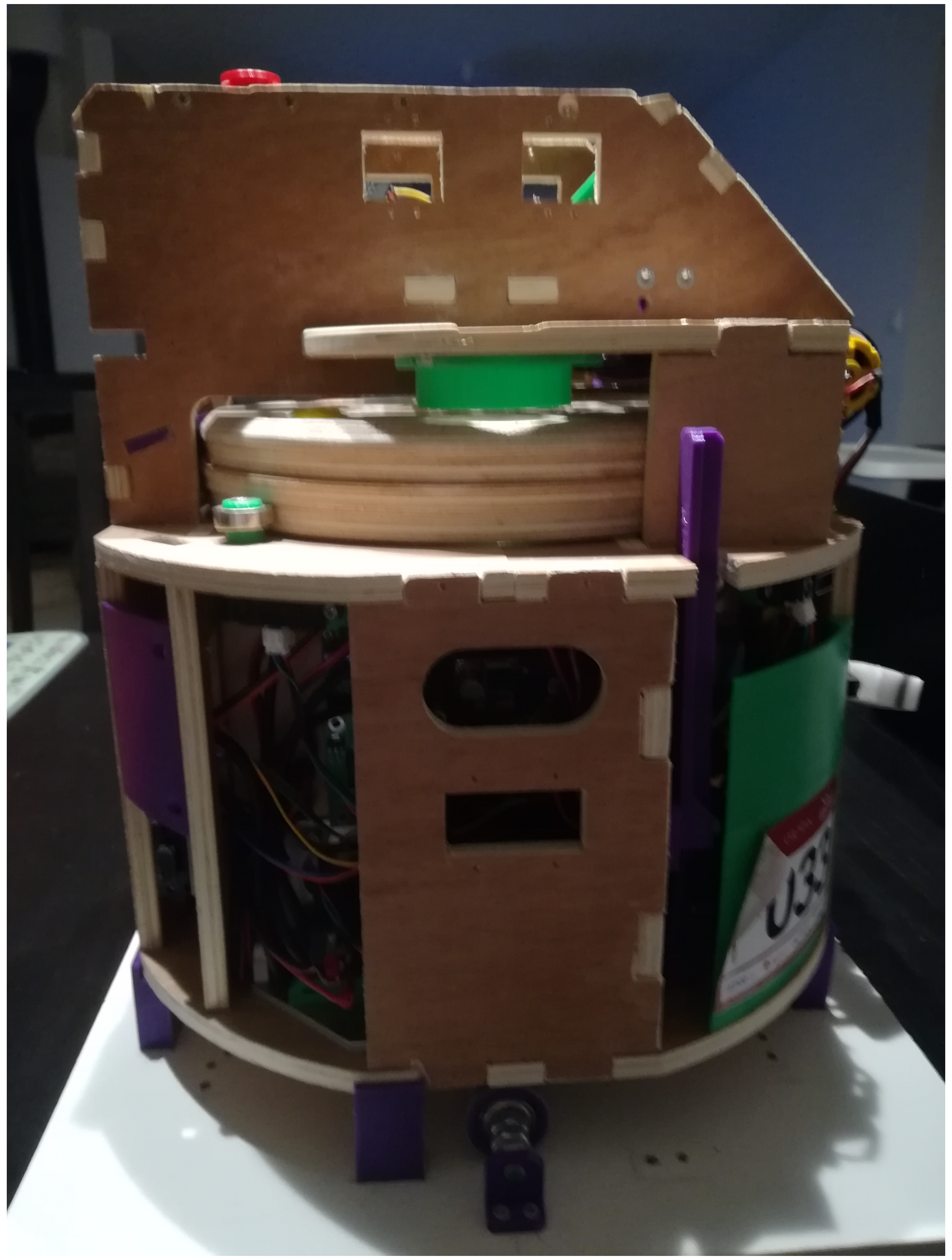
- Problems:
  - Only one robot for several developers
  - I do not run fast enough behind the robot in case of emergency :)
  - Flashing the robot several times on test table is painful
  - Robot moving is visual. How to have a visual rendering in simulation ?
- Solutions:
  - Emulate physics relation between QDEC and PWM
    - Develop PWM driver for native architecture (Incoming PR)
    - Simple average to simulate distance error between order and measure
  - Stream positional information to a 3D renderer
    - Streaming is done through console
    - Use a python script in FreeCAD parametric modeler to render robot moves

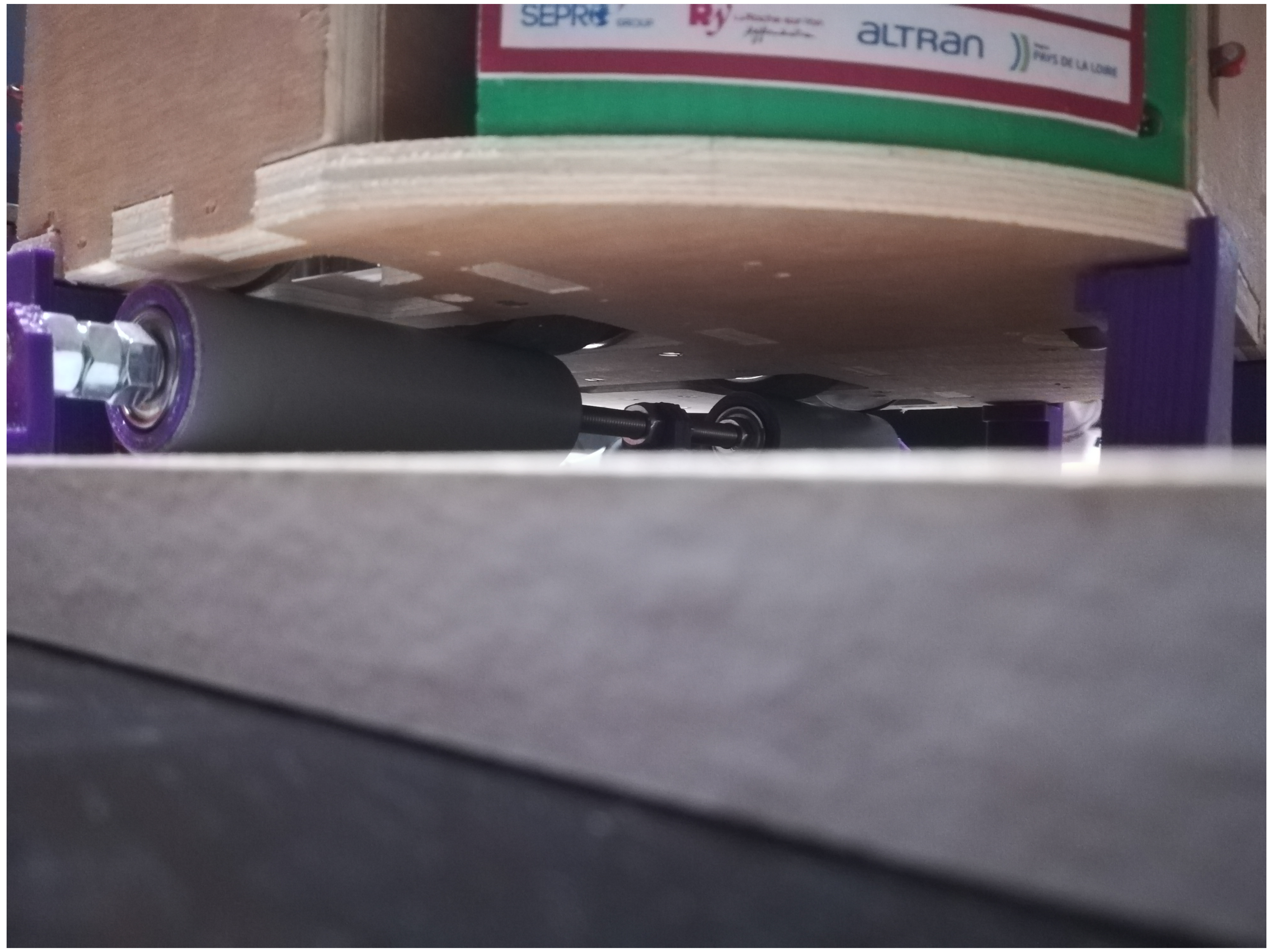
Motion control physical simulation

- Problems :
  - First physical tests can still lead to run fast to stop the robot :)
  - Context and conditions can make difficult to test the robot
- Solutions :
  - Using rollers, the robot can be tested without moving
  - Rendering is the same than for pure simulation
    - Using FreeCAD
    - Stream robot coordinates (x, y, theta) to FreeCAD through UART

Rollers







Simulation video

Final video

What’s new for 2019 ?

Incoming for 2019

- Sharp sensors are not efficient for avoidance
  - VL53L0X sensor driver
  - Neato LIDAR XV-11 driver
- Cleaning and stabilization of the source code
- Full reworking of robot scheduling
- Wireless communication (Xbee, Zigbee, …)
  - Wireless programming
  - Wireless debugging
  - Multi-Robots communication
- Testing !!!
- Sharing !!!

Useful links

- Savoir-faire Linux : https://savoirfairelinux.com/en
- Savoir-faire Linux github : https://github.com/savoirfairelinux
- COGIP : https://cogip.duckdns.org/en
- COGIP github : https://github.com/cogip
  - COGIP RIOT fork : https://github.com/cogip/RIOT
  - COGIP mcu-firmware : https://github.com/cogip/mcu-firmware

Thank you !

Questions ?

Savoir-faire Linux : gilles.doffe@savoirfairelinux.com
COGIP : cogip35@gmail.com